You will be provided with an Excel dataset containing multiple columns. Your focus will be on the "Issue" and "Sub-Issue" columns, which contain textual information describing customer issues and more specific sub-issues. Your goal is to develop an NLP model that can predict the appropriate product category to which each issue belongs.

```
import pandas as pd
import numpy as np
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
data=pd.read_csv("/content/drive/MyDrive/consumer_complaints_copy.csv")
```

```
data
```

	date_received	product	sub_product	issue	sub_issue	consumer_compl
0	12-05-2014	Debt collection	Mortgage	Disclosure verification of debt	Not given enough info to verify debt	
1	11-10-2014	Mortgage	Other mortgage	Loan servicing, payments, escrow account	NaN	
2	08/26/2015	Mortgage	Other mortgage	Loan modification,collection,foreclosure	NaN	
3	01/16/2014	Debt collection	Mortgage	Disclosure verification of debt	Not given enough info to verify debt	
4	06/25/2015	Mortgage	Conventional fixed mortgage	Application, originator, mortgage broker	NaN	My mortga misrepre
...	...	...	...	...	...	
555952	01/26/2014	Debt collection	Non-federal student loan	Improper contact or sharing of info	Contacted employer after asked not to	
555953	01/26/2016	Debt collection	Non-federal student loan	Cont'd attempts collect debt not owed	Debt was discharged in bankruptcy	
555954	03/31/2016	Debt collection	Other (i.e. phone, health club, etc.)	Disclosure verification of debt	Not given enough info to verify debt	
555955	10/13/2015	Debt collection	Credit card	Disclosure verification of debt	Not given enough info to verify debt	
555956	02-03-2014	Debt collection	Credit card	False statements or representation	Impersonated an attorney or official	

555957 rows × 18 columns

```
df=data[["issue","sub_issue","product"]]
```

```
df
```

	issue	sub_issue	product
0	Disclosure verification of debt	Not given enough info to verify debt	Debt collection
1	Loan servicing, payments, escrow account	NaN	Mortgage
2	Loan modification,collection,foreclosure	NaN	Mortgage
3	Disclosure verification of debt	Not given enough info to verify debt	Debt collection
4	Application, originator, mortgage broker	NaN	Mortgage
...	...	...	...
555952	Improper contact or sharing of info	Contacted employer after asked not to	Debt collection
555953	Cont'd attempts collect debt not owed	Debt was discharged in bankruptcy	Debt collection
555954	Disclosure verification of debt	Not given enough info to verify debt	Debt collection
555955	Disclosure verification of debt	Not given enough info to verify debt	Debt collection

```
df.shape
```

```
(555957, 3)
```

```
df.head(50)
```

10	Loan modification,collection,foreclosure	NaN	Mortgage
11	Loan modification,collection,foreclosure	NaN	Mortgage
12	Loan modification,collection,foreclosure	NaN	Mortgage
13	Application, originator, mortgage broker	NaN	Mortgage
14	Application, originator, mortgage broker	NaN	Mortgage
15	Loan modification,collection,foreclosure	NaN	Mortgage
16	Improper contact or sharing of info	Talked to a third party about my debt	Debt collection
17	Settlement process and costs	NaN	Mortgage
18	Improper contact or sharing of info	Contacted employer after asked not to	Debt collection
19	Credit decision / Underwriting	NaN	Mortgage
20	Taking/threatening an illegal action	Threatened to sue on too old debt	Debt collection
21	Credit decision / Underwriting	NaN	Mortgage
22	Loan servicing, payments, escrow account	NaN	Mortgage
23	Loan servicing, payments, escrow account	NaN	Mortgage
24	Taking/threatening an illegal action	Seized/Attempted to seize property	Debt collection
25	Problems when you are unable to pay	NaN	Consumer Loan
26	Improper contact or sharing of info	Talked to a third party about my debt	Debt collection
27	Managing the line of credit	NaN	Consumer Loan
28	Communication tactics	Frequent or repeated calls	Debt collection
29	Improper contact or sharing of info	Contacted employer after asked not to	Debt collection
30	Improper contact or sharing of info	Contacted me after I asked not to	Debt collection

```
df.sub_issue.isnull().sum()
```

```
343335
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 555957 entries, 0 to 555956
Data columns (total 3 columns):
 #   Column     Non-Null Count   Dtype 
---  ------     --------------   ----- 
 0   issue      555957 non-null  object
 1   sub_issue  212622 non-null  object
 2   product    555957 non-null  object
dtypes: object(3)
memory usage: 12.7+ MB
```

```
df.isnull().sum()
```

```
issue            0
sub_issue    343335
product           0
dtype: int64
```

```
df["product"].value_counts()
```

```
Mortgage                   186475
Debt collection            101052
Credit reporting            91854
Credit card                 66468
Bank account or service     62563
Consumer Loan               20990
Student loan                15839
Payday loan                  3877
Money transfers              3812
Prepaid card                 2470
Other financial service       557
Name: product, dtype: int64
```

```
len(df["product"].value_counts())
```

```
11
```

```
df.sub_issue.unique()
```

```
array(['Not given enough info to verify debt', nan, 'Debt is not mine',
       'Talked to a third party about my debt',
       'Contacted employer after asked not to',
       'Threatened to sue on too old debt',
       'Seized/Attempted to seize property', 'Frequent or repeated calls',
       'Contacted me after I asked not to',
       'Attempted to collect wrong amount',
       'Threatened arrest/jail if do not pay',
       'Contacted me instead of my attorney', 'Information is not mine',
       'Applied for loan/did not receive money', 'Account status',
       'Threatened to take legal action',
       'Debt was discharged in bankruptcy', "Can't contact lender",
       "Can't stop charges to bank account",
       "Received a loan I didn't apply for",
       "Charged fees or interest I didn't expect",
       'Called after sent written cease of comm',
       'Used obscene/profane/abusive language',
       'Impersonated an attorney or official',
       'Indicated committed crime not paying', 'Debt was paid',
       'Not disclosed as an attempt to collect',
       'Right to dispute notice not received',
       'Payment to acct not credited',
       'Debt resulted from identity theft', 'Account terms',
       "Sued where didn't live/sign for debt",
       'Received bad information about my loan',
       "Don't agree with fees charged",
       "Indicated shouldn't respond to lawsuit",
       'Having problems with customer service',
       'Sued w/o proper notification of suit',
       'Charged bank acct wrong day or amt', 'Repaying your loan',
       "Can't get flexible payment options",
       'Trouble with how payments are handled',
       'Need information about my balance/terms',
       "Can't temporarily postpone payments",
       "Can't decrease my monthly payments",
       'Qualify for a better loan than offered',
       'Problems when you are unable to pay', 'Getting a loan',
       'Keep getting calls about my loan',
       'Attempted to/Collected exempt funds', 'Called outside of 8am-9pm',
       'Personal information', 'No notice of investigation status/result',
       'Public record', 'Problem with statement of dispute',
       "Can't qualify for a loan", 'Billing dispute',
       'Reinserted previously deleted info',
       'Report improperly shared by CRC',
       'Problem getting report or credit score',
       'Account terms and changes', 'Inadequate help over the phone',
       'Problem cancelling or closing account',
       'Investigation took too long', 'Problem with fraud alerts',
       'Problem getting my free annual report',
       'Receiving unwanted marketing/advertising',
       'Received marketing offer after opted out',
       'Report shared with employer w/o consent', 'Insurance terms'],
      dtype=object)
```

```
df.sub_issue.value_counts()
```

```
Account status                              26798
Debt is not mine                            26285
Information is not mine                     19900
Not given enough info to verify debt        12496
Debt was paid                               11328
                                            ...
Receiving unwanted marketing/advertising      166
Report shared with employer w/o consent       127
Received marketing offer after opted out      125
Qualify for a better loan than offered        107
Insurance terms                                 4
Name: sub_issue, Length: 68, dtype: int64
```

```
len(df.sub_issue.value_counts())
```

```
68
```

```
df.issue.value_counts()
```

```
Loan modification,collection,foreclosure    97191
Incorrect information on credit report      66718
Loan servicing, payments, escrow account    60375
Cont'd attempts collect debt not owed       42285
```

```
Account opening, closing, or management     26661
                                            ...
Lost or stolen money order                     25
Incorrect exchange rate                        16
Lender damaged or destroyed vehicle             5
Lender sold the property                        5
Lender damaged or destroyed property            1
Name: issue, Length: 95, dtype: int64
```

```
len(df.issue.value_counts())
```

```
95
```

```
df["product"].unique()
```

```
array(['Debt collection', 'Mortgage', 'Consumer Loan',
       'Bank account or service', 'Credit reporting', 'Payday loan',
       'Other financial service', 'Student loan', 'Money transfers',
       'Prepaid card', 'Credit card'], dtype=object)
```

```
df["sub_issue"]=df["sub_issue"].replace(np.nan," ")
```

```
<ipython-input-20-30fe619127b1>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
  df["sub_issue"]=df["sub_issue"].replace(np.nan," ")
```

```
df
```

	issue	sub_issue	product
0	Disclosure verification of debt	Not given enough info to verify debt	Debt collection
1	Loan servicing, payments, escrow account		Mortgage
2	Loan modification,collection,foreclosure		Mortgage
3	Disclosure verification of debt	Not given enough info to verify debt	Debt collection
4	Application, originator, mortgage broker		Mortgage
...	...	...	...
555952	Improper contact or sharing of info	Contacted employer after asked not to	Debt collection
555953	Cont'd attempts collect debt not owed	Debt was discharged in bankruptcy	Debt collection
555954	Disclosure verification of debt	Not given enough info to verify debt	Debt collection
555955	Disclosure verification of debt	Not given enough info to verify debt	Debt collection
555956	False statements or representation	Impersonated an attorney or official	Debt collection

555957 rows × 3 columns

```
df["Main_issue"] = df['issue'].astype(str) +" "+ df["sub_issue"]
```

```
<ipython-input-22-3ba8f2eb55d0>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
  df["Main_issue"] = df['issue'].astype(str) +" "+ df["sub_issue"]
```

```
df
```

	issue	sub_issue	product	Main_issue
0	Disclosure verification of debt	Not given enough info to verify debt	Debt collection	Disclosure verification of debt Not given enou...
1	Loan servicing, payments, escrow account		Mortgage	Loan servicing, payments, escrow account
2	Loan modification,collection,foreclosure		Mortgage	Loan modification,collection,foreclosure
3	Disclosure verification of debt	Not given enough info to verify debt	Debt collection	Disclosure verification of debt Not given enou...
4	Application, originator, mortgage broker		Mortgage	Application, originator, mortgage broker
...	...	...	...	...
555952	Improper contact or sharing of info	Contacted employer after asked not to	Debt collection	Improper contact or sharing of info Contacted ...

```
df1=df[["Main_issue","product"]]
```

owed in bankruptcy collection Debt was...

```
df1
```

	Main_issue	product
0	Disclosure verification of debt Not given enou...	Debt collection
1	Loan servicing, payments, escrow account	Mortgage
2	Loan modification,collection,foreclosure	Mortgage
3	Disclosure verification of debt Not given enou...	Debt collection
4	Application, originator, mortgage broker	Mortgage
...	...	...
555952	Improper contact or sharing of info Contacted ...	Debt collection
555953	Cont'd attempts collect debt not owed Debt was...	Debt collection
555954	Disclosure verification of debt Not given enou...	Debt collection
555955	Disclosure verification of debt Not given enou...	Debt collection
555956	False statements or representation Impersonate...	Debt collection

555957 rows × 2 columns

```
# for importing label encoder
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
# for applying label encoder
df1['product'] = le.fit_transform(df['product'])
```

```
<ipython-input-27-f070be819bc9>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
  df1['product'] = le.fit_transform(df['product'])
```

```
df1
```

	Main_issue	product
0	Disclosure verification of debt Not given enou...	4
1	Loan servicing, payments, escrow account	6
2	Loan modification,collection,foreclosure	6
3	Disclosure verification of debt Not given enou...	4
4	Application, originator, mortgage broker	6

```
# X for input y for output/ prediction
X = df1['Main_issue']
y = df1['product']
```

555953	Cont'd attempts collect debt not owed Debt was...	4

```
X
```

```
0         Disclosure verification of debt Not given enou...
1                Loan servicing, payments, escrow account
2                Loan modification,collection,foreclosure
3         Disclosure verification of debt Not given enou...
4                Application, originator, mortgage broker
                              ...
555952    Improper contact or sharing of info Contacted ...
555953    Cont'd attempts collect debt not owed Debt was...
555954    Disclosure verification of debt Not given enou...
555955    Disclosure verification of debt Not given enou...
555956    False statements or representation Impersonate...
Name: Main_issue, Length: 555957, dtype: object
```

```
y
```

```
0         4
1         6
2         6
3         4
4         6
         ..
555952    4
555953    4
555954    4
555955    4
555956    4
Name: product, Length: 555957, dtype: int64
```

```
y.value_counts()
```

```
6     186475
4     101052
3      91854
2      66468
0      62563
1      20990
10     15839
8       3877
5       3812
9       2470
7        557
Name: product, dtype: int64
```

```
# for stlyling only
from tqdm import tqdm
import nltk
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
import re
```

```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
ps = PorterStemmer()
corpus = []
```

```
for i in tqdm(range(len(X))):
#     print(i, end=', ')
    review = re.sub("[^a-zA-Z]"," ",X[i])
    review = review.lower()
    review = review.split()
    review = [ps.stem(word) for word in review if word not in set(stopwords.words("english"))]
    review = " ".join(review)
    corpus.append(review)
```

```
 100%|██████████| 555957/555957 [08:22<00:00, 1105.86it/s]
```

```
#we get a clean text



# for feature extraction
from sklearn.feature_extraction.text import TfidfVectorizer
cv = TfidfVectorizer()

X = cv.fit_transform(corpus).toarray()


X
```

```
array([[0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.38862285, 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       ...,
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ]])
```

```
X.shape
```

```
(555957, 232)
```

```
# splitting
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, random_state=81)


# svm
from sklearn.svm import SVC
model = SVC(C=1.0,
            kernel='linear',
            random_state=10)


# fit
model.fit(X_train, y_train)
```

```
SVC(kernel='linear', random_state=10)
```

```
model.score(X_test, y_test)
```

```
0.9836858766817757
```

```
# predict
y_pred = model.predict(X_test)


from sklearn.metrics import classification_report
print(classification_report(y_pred,y_test))
```

```
                   precision    recall  f1-score   support

               0       1.00      0.99      0.99     12680
               1       0.99      0.92      0.95      4555
               2       1.00      0.92      0.96     14444
```

```
                3       1.00      1.00      1.00     18330
                4       1.00      1.00      1.00     20078
                5       0.76      0.84      0.80       692
                6       0.98      1.00      0.99     36489
                7       0.33      1.00      0.50        36
                8       1.00      0.93      0.97       825
                9       0.46      1.00      0.63       225
               10       0.89      1.00      0.94      2838

         accuracy                           0.98    111192
        macro avg       0.85      0.96      0.88    111192
     weighted avg       0.99      0.98      0.98    111192
```

```
# Assuming you have X_test, y_test, and y_pred arrays as NumPy arrays or lists

# Create a DataFrame with X_test, y_test, and y_pred
d = {'X_test': X_test.tolist(), 'y_test': y_test.tolist(), 'y_pred': y_pred.tolist()}
df2 = pd.DataFrame(d)

# Display the DataFrame
print(df2)
```

```
                                                  X_test  y_test  y_pred
0       [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...       2       2
1       [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...      10      10
2       [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...       6       6
3       [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...       2       2
4       [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...       1       1
...                                                   ...     ...     ...
111187  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...       4       4
111188  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...       6       6
111189  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...       4       4
111190  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...       1       1
111191  [0.0, 0.3886228528716477, 0.0, 0.0, 0.0, 0.0, ...       6       6

[111192 rows x 3 columns]
```

```
import pickle


# dump the preprocessing file
pickle.dump(cv, open('customer_Main_issue_cv.pkl', 'wb'))  # write binary
# dump the model
pickle.dump(model, open('customer_Main_issue_model.pkl', 'wb'))


save_cv = pickle.load(open('customer_Main_issue_cv.pkl', 'rb'))  # read binary
save_model = pickle.load( open('customer_Main_issue_model.pkl', 'rb'))


# function for testing

def test_model(sentence):
    sen = save_cv.transform([sentence]).toarray()

    res = save_model.predict(sen)[0]

    if res == 0:
        print('Bank account or service')
    elif res == 1:
        print('customer loan')
    elif res == 2:
        print('Credit Card')
    elif res == 3:
        print('Creadit Reporting')
    elif res == 4:
        print('Debt Collection')
    elif res == 5:
        print('Money Transfer')
    elif res == 6:
        print('Mortgage')
    elif res == 7:
        print('Other Financial Service')
    elif res == 8:
        print('PayDay Loan')
    elif res == 9:
        print('Prepaid card')
```